FerriteHyperelastic.jl: A Julia Package for Finite Element Analysis of Hyperelastic Materials

Amin Alibakhshi[a*], Kian Aghani [b*], Luis Saucedo-Mora [a+]

[a] *E.T.S. de Ingeniería Aeronáutica y del Espacio, Universidad Politécnica de Madrid, Pza. Cardenal Cisneros 3, 28040, Madrid, Spain*

[b] *Research assistant, Sahand University of Technology, Tabriz, Iran*

[*] *Amin Alibakhshi and Kian Aghani contributed equally to this work.*

[+] *Corresponding author: luis.saucedo@upm.es*

Abstract

Nonlinear analysis of hyperelastic materials presents unique challenges in finite element (FE) simulations, particularly in achieving efficient implementation of finite deformation analysis and providing accessible tools for material parameter identification. This article provides FerriteHyperelastic.jl, a comprehensive open-source package for FE analysis of hyperelastic materials, built upon the robust Ferrite.jl framework in Julia programming language. FerriteHyperelastic.jl supports both two- and three-dimensional finite element formulations for hyperelastic materials. A feature of the package is its integrated curve-fitting module, employing numerical procedures to determine material constants directly from uniaxial, biaxial, and pure shear experimental data, while performing stability tests. A comprehensive documentation, encompassing formulations and practical examples is also presented. FerriteHyperelastic.jl provides a distinguished contribution to the FE community, providing an extensible and efficient platform for analysis of hyperelastic structures. Visualization of the geometry, mesh, boundary conditions, and deformation is also provided in the package.

Keywords: finite element, hyperelasticity, julia language, curve fitting, plane stress deformation

1. Motivation and significance

The efficient implementation of the finite element (FE) for hyperelastic structures presents theoretical and computational challenges, particularly in two-dimensional plane stress deformation, in which the out-of-plane stress component in zero while the corresponding strain remains unknown [1]. Coupled with the nonlinear behavior of hyperelastic materials, this constraint substantially increases computational complexity. The accurate modeling of hyperelastic materials is vital for engineering applications, making efficient and accessible computational tools crucial for research purposes and industrial applications. The finite element software for non-linear analysis is dominated by two categories: commercial packages and open-source implementations. Commercial software offers robust hyperelasticity modules; however, these solutions present significant barriers including closed-source architectures and substantial licensing costs. On the other hand, open-source packages target research purposes and commonly written in C, C++, or FORTRAN, exemplified by libraries such as FEBio [2]. Although these packages offer

efficiency and accessibility, they suffer from steep learning curves and verbose syntax that obscures the elegance of FE procedure.

Computational mechanics software development has experienced advancements by the emergence of high-level computing languages such a Python. Software packages such as FEniCS [3] cover both linear and nonlinear finite element analysis (FEA). However, Python's interpreted nature introduces performance limitations that become an issue in simulations having high number of degrees of freedom (DOF). This matter becomes more pronounced in element-level operations demanding nested loops .

By the introduction of Julia programming language in 2012 [4], a compelling solution to the "two-language problem" in scientific computing is given by an innovative design combining multiple dispatch, just-in-time compilation, meta-programming capabilities, and optional type annotations. Moreover, the language's syntax closely mirrors mathematical notation, paving the way for direct translation of FE formulations. Within the Julia FE ecosystem, two packages have emerged as foundational frameworks: Ferrite.jl [5] and Gridap.jl [6]. Ferrite.jl offers an explicit framework, providing control over the FE assembly loop by prioritizing flexibility and extensibility. It allows for the straightforward implementation of complex constitutive models and arbitrary element formulations, making it an ideal foundation for researchers. In contrast, Gridap.jl adopts a more declarative framework. By employing functional programming paradigms, it provides a highly expressive methodology for defining and solving partial differential equations. While both packages provide significant advancements for FEA in Julia, their native support for the domain of hyperelasticity remains limited, which reveals several critical gaps for a researcher or engineer. Critical gaps include absence of specialized plane stress algorithms, lack of integrated parameter identification tools, and minimal support for common experimental configurations such as uniaxial, biaxial, and pure shear tests.

In this work, we present FerriteHyperelastic.jl, a specialized Julia package that extends Ferrite.jl with comprehensive capabilities for FEA of hyperelastic materials. Our package addresses the aforementioned limitations through several key contributions: (1) integrated curve-fitting modules for material parameter identification and stability check from uniaxial, biaxial, and pure shear experimental data for Neo-Hookean, Mooney-Rivlin, Yeoh, and Ogden models; (2) robust plane stress algorithms employing local Newton iterations with consistent linearization; (3) specialized element formulations (hybrid method) for nearly incompressible materials including mixed methods; (4) visualization tools for pre- and post-processing. The package provides an advancement in accessible, efficient, and mathematically transparent tools for hyperelastic finite element analysis. The package is open-access and freely available at https://github.com/Aminofa70/FerriteHyperelastic.jl.

2. Software description

FerriteHyperelastic.jl can be used for the FEA of two- and three-dimensional hyperelastic structures. Thanks to the capabilities of Ferrite.jl, in FerriteHyperelastic, different mesh types and resolutions can be implemented, Neuman and Dirichlet boundary conditions are specified easily, and results are saved in a VTU format for post-processing. Furthermore, because Ferrite.jl supports automatic differentiation, many hyperelastic models can be easily implemented in a small number of lines of code, and related stresses and their differentiation are obtained. It also encompasses the curve-fitting for defining the material parameters in hyperelastic models used in the FEM.

2.1. Software architecture

FerriteHyperelastic.jl is written in the Julia language and contains different folders. The main folder is src, which includes the module and all source functions. Moreover, the package has the folder examples that provide different 2D and 3D problems with different hyperelastic models as a starting point.

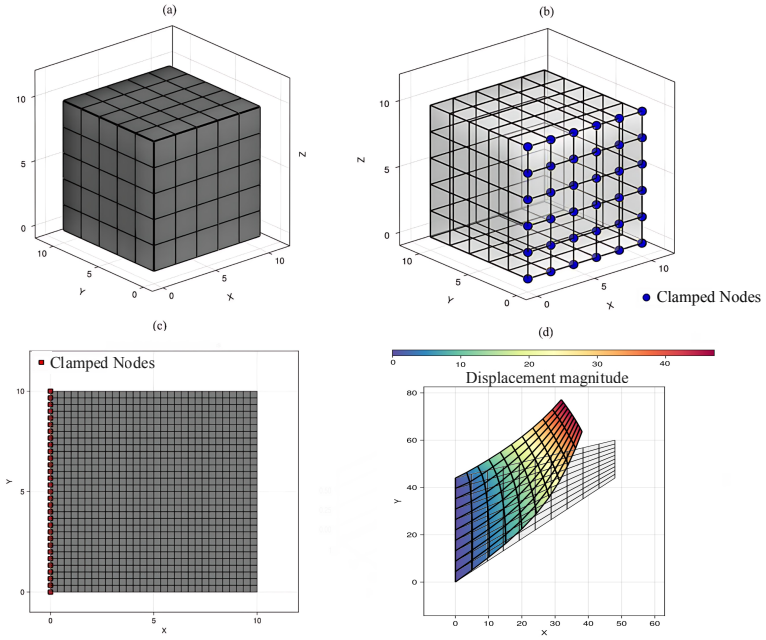

Figure 1: An example of the geometry, mesh, boundary conditions, and displacement field generated by the built-in functions in the package. (a) Mesh and geometry of the hexahedron element, (b) Boundary condition in the hexahedron mesh, (c) Boundary condition, geometry, and mesh of the quadrilateral element, and (d) Displacement field in the hexahedron element.

2.2. Software functionalities

The package provides many functions for better and more detailed performance. It performs FEA for 2D plane stress and strain and 3D cases with different hyperelastic models. It also supplies a curve-fitting tool to find the value of material parameters in hyperelastic models as well as the stability conditions. Although the results are saved in VTU format and post-processing can be done in open-access software like ParaView [7], plots of the mesh, showing boundary conditions, and results are also included. Figure 1 presents the mesh, boundary conditions, and displacement for two- and three-dimensional structures generated by the package. For the plot, we follow the style used in the notable packages GIBBON [8] and Comodo.jl [9].

2.3. Execution of the code

To execute the code and perform FEA, the Julia language and FerriteHyperelastic.jl should be installed. To execute the code and perform FEA, the Julia language and FerriteHyperelastic.jl should be installed. The package is built on top of other Julia packages, mainly Ferrite.jl and others such as Makie.jl [10] (using its backend GLMakie), GeometryBasics.jl [11], and Tensors.jl [12].

2.4. Solver properties

FerriteHyperelastic.jl uses a robust automatic time-integration scheme to efficiently handle the non-linearities inherent to the analysis. This strategy is formulated to balance computational cost with solution stability. The analysis initializes with an initial increment size (L_{init}). Upon achieving convergence for a given increment, the solver increases the size of the subsequent increment. This accelerates the solution process when the material response is relatively stable. This growth is constrained by the maximum increment size (L_{max}).

If the equilibrium iterations for an increment fail to converge, the solver automatically performs a cutback, reducing the increment size and re-attempts the solution. This process is repeated until convergence is achieved; however, the increment size will not be reduced below a specified minimum threshold (L_{min}). While the user can configure these parameters, the library assumes default values for L_{init}, L_{min}, and L_{max} as 0.1, 10^{-5}, and 0.1, respectively, in the absence of input values.

3. Illustrative examples

This section outlines the package's capabilities in curve fitting and finite element analysis.

3.1. Curve fitting

The curve fitting is conducted for the following models [13, 14] to predict the experimental data of Treloar in [15]

$$\begin{aligned} W_{\text{Neo-Hookean}} &= C_{10}(\bar{I}_1 - 3) \\ W_{\text{Mooney-Rivlin}} &= C_{10}(\bar{I}_1 - 3) + C_{01}(\bar{I}_2 - 3) \\ W_{\text{Yeoh}} &= C_{10}(\bar{I}_1 - 3) + C_{20}(\bar{I}_1 - 3)^2 + C_{30}(\bar{I}_1 - 3)^3 \\ W_{\text{Ogden}} &= \sum_{p=1}^{N} \frac{2\mu_p}{\alpha_p^2} \left(\lambda_1^{\alpha_p} + \lambda_2^{\alpha_p} + \lambda_3^{\alpha_p} - 3\right) \end{aligned} \tag{1}$$

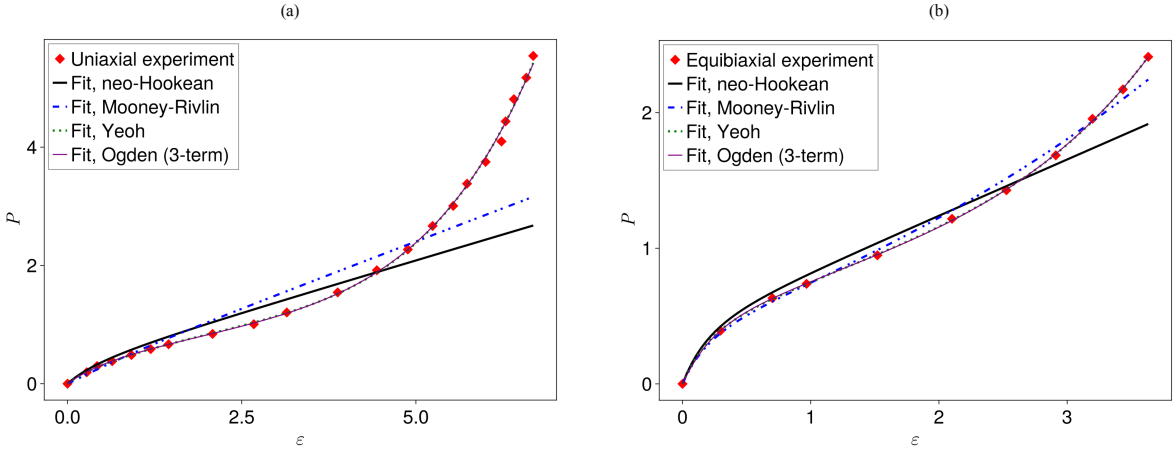

Figure 2: Fitting hyperelastic models with experimental data of Treloar [15]. (a) Uniaxial test, and (b) Equibiaxial test.

In addition to finding the material constants, the Drucker stability criterion [14] is also implemented. Figure 2 shows the accuracy of different models in fitting Treloar's test data.

3.2. Finite element analysis

In this section, we show the efficiency and accuracy of the proposed package through its implementation for three structures. The hyperelastic model here is neo-Hookean, as [14]

$$W = C_{10}(\bar{I}_1 - 3) + 1/D_1(J-1)^2 \tag{2}$$

Figure 3(a) is a two-dimensional plane stress plate with the clamped edge on the left and the right edge subjected to a traction load. Figure 3(b) illustrates the Cook's membrane in the plane strain mode, and Figure 3(c) is a three-dimensional structure with shown boundary conditions where the traction is in the positive direction of the x-axis. The result of the package is compared with a MATLAB code for the finite element analysis presented in [14].

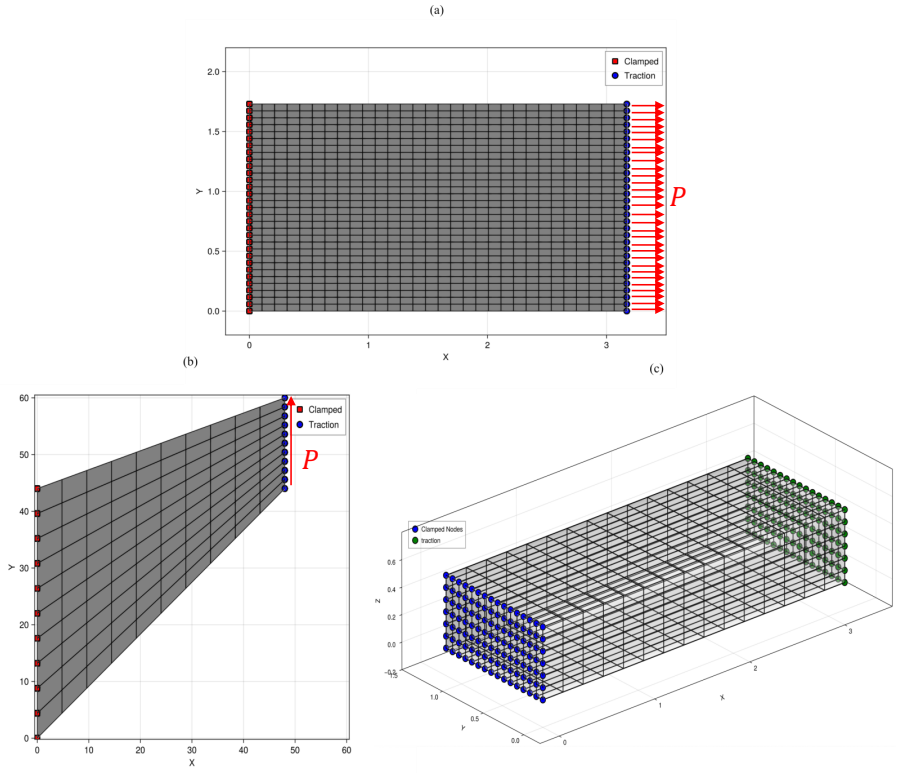

Figure 3: The geometry, applied loads, and boundary conditions relevant to the problems under analysis. (a) Clamped plate subjected to a tension load, (b) Cook's cantilever, and (c) Clamped beam in response to a tension load.

As shown in Table 1, the maximum displacement in x and y direction obtained from the Julia and MATLAB codes is shown. It is observed that the difference between the two codes is very small, ensuring the accuracy of the results derived from FerriteHyperelastic.jl.

Table 1: FEM Results Comparison: Julia vs MATLAB

Problem	**FEM Inputs**	**Load**	**Julia**	**MATLAB**
Plane tension (plane strain)	Nx = 30, Ny = 30 E = 4.35, ν = 0.45	P = 0.17	max(ux) = 0.097003 max(uy) = 0.022423	max(ux) = 0.09708 max(uy) = 0.022441
		P = 1.17	max(ux) = 0.799932 max(uy) = 0.155599	max(ux) = 0.80539 max(uy) = 0.15646
Cook membrane (plane stress)	Nx = 10, Ny = 10 E = 4.35, ν = 0.45	P = 0.17	max(ux) = 0.792604 max(uy) = 11.676090	max(ux) = 0.7923 max(uy) = 11.6937
		P = 1.17	max(ux) = 1.5999271 max(uy) = 39.2521	max(ux) = 1.6004 max(uy) = 39.3683
Beam three dimensional	Nx = 15, Ny = 15, Nz = 6 E = 20, ν = 0.45	P = 0.2	max(ux) = 0.0307723 max(uy) = 0.0027103	max(ux) = 0.030839 max(uy) = 0.0027161
		P = 1.0	max(ux) = 0.1577365 max(uy) = 0.01345641	max(ux) = 0.15949 max(uy) = 0.0136

In Figure 4, we show the force-displacement plot for the neo-Hookean model. We apply the finite element analysis for a plate in plane strain deformation state and subjected to a uniaxial tension. As we see, the package captures the nonlinearity of the model.

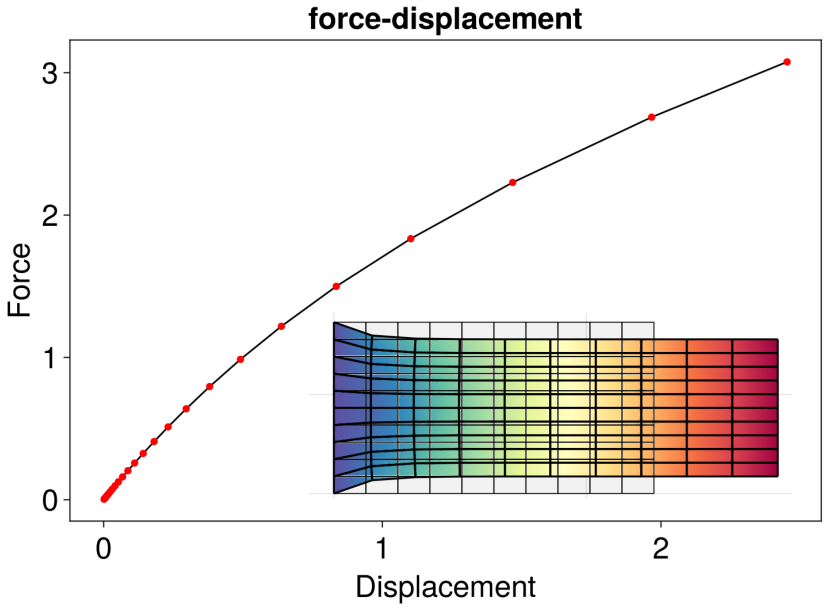

Figure 4: Force-displacement curve for the neo-Hookean model.

4. Conclusions

In this paper, we introduced a Julia package for the finite element analysis of hyperelastic materials. The package covers two and three-dimensional problems. It also provides curve fitting with stability for some hyperelastic models, such as the neo-Hookean, Mooney-Rivlin, Yeoh, and Ogden models. The results, mesh, and boundary conditions can be visualized using built-in functions in the package. Additionally, they can be saved as a VTU file for use in other software such as ParaView. The package is a plugin for Ferrite.jl that includes many capabilities, such as defining different mesh types, quadrature points, and interpolations.

Author Contributions:

A. Alibakhshi: Conceptualization, Data curation, Investigation, Visualization, Writing – original draft, Software. **K. Aghani** Conceptualization, Data curation, Investigation, Visualization, Writing – review and editing. **L. Saucedo-Mora**: Project administration, Supervision, Validation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Masoud Ahmadi, Andrew McBride, Paul Steinmann, and Prashant Saxena. Plane stress finite element modelling of arbitrary compressible hyperelastic materials: M. ahmadi et al. *Acta Mechanica*, pages 1–20, 2025.

[2] Steve A Maas, Benjamin J Ellis, Gerard A Ateshian, and Jeffrey A Weiss. Febio: finite elements for biomechanics. *Journal of Biomechanical Engineering*, 2012.

[3] I. A. Baratta, J. P. Dean, J. S. Dokken, M. Habera, J. S. Hale, C. N. Richardson, M. E. Rognes, M. W. Scroggs, N. Sime, and G. N. Wells. DOLFINx: The next generation FEniCS problem solving environment. preprint, 2023.

[4] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B Shah. Julia: A fresh approach to numerical computing. *SIAM review*, 59(1):65–98, 2017.

[5] K. Carlsson, F. Ekre, and Ferrite.jl contributors. Ferrite.jl (v1.1.0), 2025.

[6] Santiago Badia and Francesc Verdugo. Gridap: An extensible finite element toolbox in julia. *Journal of Open Source Software*, 5(52):2520, 2020.

[7] James Ahrens, Berk Geveci, and Charles Law. Paraview: An end-user tool for large data visualization. *The visualization handbook*, 717(8), 2005.

[8] Kevin M Moerman. Gibbon: the geometry and image-based bioengineering add-on. *Journal of Open Source Software*, 3(22):506, 2018.

[9] Kevin Mattheus Moerman, Mehmet Hakan Satman, Chethana Rao, Daniel Vanden-Heuvel, Juan Ignacio Polanco, and Simon Danish. Comodo.jl: A Julia Package for Computational Mechanics and Design (v1.0.0), 2025.

[10] Simon Danisch and Julius Krumbiegel. Makie. jl: Flexible high-performance data visualization for julia. *Journal of Open Source Software*, 6(65):3349, 2021.

[11] Arsh Sharma Simon, Anshul Singhvi, Jeremie Knuesel, Steve Kelly, Martijn Visser, Frederic Freyer, Tim Holy, Rafael Schouten, Maarten Pronk, Pietro Vertechi, Oliver Evans, Tianyi Pu, Júlio Hoffimann, Jan Weidner, Daniel Schwabeneder, Alex Hirzel, Hendrik Ranocha, mtsch, and Yuto Horikawa. JuliaGeometry/GeometryBasics.jl: v0.5.9, 2025.

[12] Kristoffer Carlsson and Fredrik Ekre. Tensors. jl—tensor computations in julia. *Journal of Open Research Software*, 7(1), 2019.

[13] Po-Sen Lin, Olivier Le Roux de Bretagne, Marzio Grasso, James Brighton, Chris StLeger-Harris, and Owen Carless. Comparative analysis of various hyperelastic models and element types for finite element analysis. *Designs*, 7(6):135, 2023.

[14] Salar Farahmand-Tabar and Kian Aghani. *Practical Programming of Finite Element Procedures for Solids and Structures with MATLAB®: From Elasticity to Plasticity.* Elsevier, 2023.

[15] Leslie RG Treloar. Stress-strain data for vulcanized rubber under various types of deformation. *Rubber Chemistry and Technology*, 17(4):813–825, 1944.